variables and arithmetic expressions in python.

Goal: Learn about variables and arithmetic expressions in python.

A simple program can best illustrate the use of arithmetic expressions.

```
# simple_program.py
pi = 3.1415
degs = 0
while degs <= 90:
 rads = degs * pi / 180.0
 print degs, " degrees = ", rads, " radians"
 degs = degs + 10.
```

The output of this program is the following table:

```
0 degrees = 0.0 radians
10.0 degrees = 0.174527777778 radians
20.0 degrees = 0.349055555556 radians
30.0 degrees = 0.523583333333 radians
40.0 degrees = 0.698111111111 radians
50.0 degrees = 0.872638888889 radians
60.0 degrees = 1.04716666667 radians
70.0 degrees = 1.22169444444 radians
80.0 degrees = 1.39622222222 radians
90.0 degrees = 1.57075 radians
```

Python is a dynamically typed language (i.e you do not need to declare the variable types apriori) and the names can represent different types depending on the arithmetic operations performed. In the above example, "degs" was initially set to the integer 0, and is seen in the first line printed. Subsequently a real number (10.) is added at each stage of the loop and this changed the type of "degs" as it can be seen that "degs" values printed are real. Note that the above example is contrived to illustrate this point and there is no real reason to add a real value 10.0 in the loop.

The output of the program looks less than ideally formatted. To make it look better, we can make use of format strings. For example:

```
>>> print "%3d degrees = %0.4f radians" %(degs, rads)
```

would produce output that looks like this:

```
 0 degrees = 0.0000 radians
10 degrees = 0.1745 radians
20 degrees = 0.3491 radians
30 degrees = 0.5236 radians
40 degrees = 0.6981 radians
50 degrees = 0.8726 radians
60 degrees = 1.0472 radians
70 degrees = 1.2217 radians
80 degrees = 1.3962 radians
90 degrees = 1.5708 radians
```

The format strings ***%d***, ***%s*** denote integers and strings respectively, and ***%g*** and ***%f*** denote floats.

NOTE:
When a command is split among several lines with the line−continuation marker ("\"), the continued lines can be indented in any manner; Python's normally stringent indentation rules do not apply.